8-2020

Hardness of Sparse Sets and Minimal Circuit Size Problem

Bin Fu
The University of Texas Rio Grande Valley

Recommended Citation

Fu B. (2020) Hardness of Sparse Sets and Minimal Circuit Size Problem. In: Kim D., Uma R., Cai Z., Lee D. (eds) Computing and Combinatorics. COCOON 2020. Lecture Notes in Computer Science, vol 12273. Springer, Cham. https://doi.org/10.1007/978-3-030-58150-3_39

Hardness of Sparse Sets and Minimal Circuit Size Problem

Bin Fu
Department of Computer Science,
University of Texas Rio Grande Valley, Edinburg, TX 78539, USA.
bin.fu@utrgv.edu

Abstract

We study the magnification of hardness of sparse sets in nondeterministic time complexity classes on a randomized streaming model. One of our results shows that if there exists a $2^{n^{o(1)}}$-sparse set in NTIME($2^{n^{o(1)}}$) that does not have any randomized streaming algorithm with $n^{o(1)}$ updating time, and $n^{o(1)}$ space, then NEXP $\neq$ BPP, where a $f(n)$-sparse set is a language that has at most $f(n)$ strings of length n. We also show that if MCSP is ZPP-hard under polynomial time truth-table reductions, then EXP $\neq$ ZPP.

1. Introduction

Hardness magnification has been intensively studied in the recent years [13, 4, 10, 12]. A small lower bound such as $\Omega(n^{1+\epsilon})$ for one problem may bring a large lower bound such as super-polynomial lower bound for another problem. This research is closely related to Minimum Circuit Size Problem (MCSP) that is to determine if a given string of length $n = 2^m$ with integer m can be generated by a circuit of size k. For a function $s(n) : \mathbb{N} \to \mathbb{N}$, MCSP[$s(n)$] is that given a string x of length $n = 2^m$, determine if there is a circuit of size at most $s(n)$ to generate x. This problem has received much attention in the recent years [3, 2, 13, 8, 7, 6, 5, 4, 12, 11, 10].

Hardness magnification results are shown in a series of recent papers about MCSP [13, 4, 10, 12]. Oliveira and Santhanam [13] show that $n^{1+\epsilon}$-size lower bounds for approximating MKtP[n^{β}] with an additive error $O(\log n)$ implies EXP $\not\subseteq$ P/poly. Oliveira, Pich and Santhanam [12] show that for all small $\beta > 0$, $n^{1+\epsilon}$-size lower bounds for approximating MCSP[$n^{\beta m}$] with factor $O(m)$ error implies NP $\not\subseteq$ P/poly. McKay, Murray, and Williams [10] show that an $\Omega(n\text{poly}(\log n))$ lower bound on poly($\log n$) space deterministic streaming model for MCSP[poly($\log n$)] implies separation of P from NP.

The hardness magnification of non-uniform complexity for sparse sets is recently developed by Chen, Jin and Williams [4]. Since MCSP[$s(n)$] are of sub-exponential density for $s(n) = n^{o(1)}$, the hardness magnification for sub-exponential density sets is more general than the hardness magnification for MCSP. They show that if there is an $\epsilon > 0$ and a family of languages $\{L_b\}$ (indexed over $b \in (0,1)$) such that each L_b is a 2^{n^b}-sparse language in NP, and $L_b \notin$ Circuit$[n^{1+\epsilon}]$, then NP $\not\subseteq$ Circuit$[n^k]$ for all k, where Circuit$[f(n)]$ is the class of languages with nonuniform circuits of size bounded by function $f(n)$. Their result also holds for all complexity classes C with $\exists C = C$.

On the other hand, it is unknown if MCSP is NP-hard. Murray and Williams [11] show that NP-completeness of MCSP implies the separation of EXP from ZPP, a long standing unsolved problem in computational complexity theory. Hitchcock and Pavan [8, 11] if MCSP is NP-hard under polynomial time truth-table reductions, then EXP$\not\subseteq$ NP $\cap$ P/poly.

Separating NEXP from BPP, and EXP from ZPP are two of major open problems in the computational complexity theory. We are motivated by further relationship about sparse sets and MCSP, and the two separations NEXP $\neq$ BPP and EXP $\neq$ ZPP. We develop a polynomial method on finite fields to magnify the hardness of sparse sets in nondeterministic time complexity classes over a randomized streaming model. One of our results show that if there exists a $2^{n^{o(1)}}$-sparse set in NTIME($2^{n^{o(1)}}$) that does not have a randomized streaming algorithm with $n^{o(1)}$ updating time, and $n^{o(1)}$ space, then NEXP $\neq$ BPP, where a $f(n)$-sparse set is a language that has at most $f(n)$ strings of length n. Our magnification result has a flexible trade off between the spareness and time complexity.

We use two functions $d(n)$ and $g(n)$ to control the sparseness of a tally set T. Function $d(n)$ gives an upper bound for the number of elements of in T and $g(n)$ is the gap lower bound between a string 1^n and the next string 1^m in T, which satisfy $g(n) < m$. The class TALLY$(d(n), g(n))$ defines the class of all those tally sets. By choosing $d(n) = \log\log n$, and $g(n) = 2^{2^{2n}}$, we prove that if MCSP is ZPP $\cap$ TALLY$(d(n), g(n))$-hard under polynomial time truth-table reductions, then EXP $\neq$ ZPP.

1.1. Comparison with the existing results

Comparing with some existing results about sparse sets hardness magnification in this line [4], there are some new advancements in this paper.

1. Our magnification of sparse set is based on a uniform streaming model. A class of results in [4] are based on nonuniform models. In [10], they show that if there is $A \in$ PH, and a function $s(n) \geq \log n$, search-MCSP$^A[s(n)]$ does not have $s(n)^c$ updating time in deterministic streaming model for all positive, then P $\neq$ NP. MCSP$[s(n)]$ is a $s(n)^{\mathrm{O}(s(n))}$-sparse set.

2. Our method is conceptually simple, and easy to understand. It is a polynomial algebraic approach on finite fields.

3. A flexible trade off between sparseness and time complexity is given in our paper.

Proving NP-hardness for MCSP implies EXP $\neq$ ZPP [8, 11]. We consider the implication of ZPP-hardness for MCSP, and show that if MCSP is ZPP $\cap$ TALLY$(d(n), g(n))$-hard for a function pair such as $d(n) = \log\log n$ and $g(n) = 2^{2^{2n}}$, then EXP $\neq$ ZPP. It seems that proving MCSP is ZPP-hard is much easier than proving MCSP is NP-hard since ZPP $\subseteq$ (NP$\cap$co-NP) $\subseteq$ NP. According to the low-high hierarchy theory developed by Schöning [14], the class NP$\cap$co-NP is the low class L_1. Although MCSP may not be in the class ZPP, it is possible to be ZPP-hard.

2. Notations

Minimum Circuit Size Problem (MCSP) is that given an integer k, and a binary string T of length $n = 2^m$ for some integer $m \geq 0$, determine if T can be generated by a circuit of size k. Let $\mathbb{N} = \{1, 2, \cdots\}$ be the set of all natural numbers. For a language L, L^n is the set of strings in L of length n, and $L^{\leq n}$ is the set of strings in L of length at most n. For a finite set A, denote $|A|$ to be the number of elements in A. For a string s, denote $|s|$ to be its length. If x, y, z are not empty strings, we have a coding method that converts a x, y into a string $\langle x, y\rangle$ with $|x|+|y| \leq |\langle x, y\rangle| \leq 3(|x|+|y|)$ and converts x, y, z into $\langle x, y, z\rangle$ with $|x|+|y|+|z| \leq |\langle x, y, z\rangle| \leq 3(|x|+|y|+|z|)$. For example, for $x = x_1 \cdots x_{n_1}, y = y_1 \cdots y_{n_2}, z = z_1 \cdots z_{n_3}$, let $\langle x, y, z\rangle = 1x_1 \cdots 1x_{n_1} 001 y_1 \cdots 1y_{n_2} 001 z_1 \cdots 1z_{n_3}$.

Let DTIME$(t(n))$ be the class of languages accepted by deterministic Turing machines in time O$(t(n))$. Let NTIME$(t(n))$ be the class of languages accepted by nondeterministic Turing machines in time O$(t(n))$. Define EXP $= \cup_{c=1}^{\infty}$DTIME(2^{n^c}) and NEXP $= \cup_{c=1}^{\infty}$NTIME(2^{n^c}). P/poly, which is also called PSIZE, is the class of languages that have polynomial-size circuits.

We use a polynomial method on a finite field F. It is classical theory that each finite field is of size p^k for some prime number p and integer $k \geq 1$ (see [9]). For a finite field F, we denote $R(F) = (p, t_F(u))$ to represent F, where $t_F(u)$ is a irreducible polynomial over field GF(p) for the prime number p and its degree is $\deg(t_F(.)) = k$. The polynomial $t_F(u)$ is equal to u if F is of size p, which is a prime number. Each element of F with $R(F) = (p, t_F(u))$ is a polynomial $q(u)$ with degree less than the degree of $t_F(u)$. For two elements $q_1(u)$ and $q_2(u)$ in F, their addition is defined by $(q_1(u) + q_2(u))(\text{mod } t_F(u))$, and their multiplication is defined by $(q_1(u) \cdot q_2(u))(\text{mod } t_F(u))$ (see [9]). Each element in GF(2^k) is a polynomial $\sum_{i=0}^{k-1} b_i u^i$ ($b_i \in \{0, 1\}$), which is represented by a binary string $b_{k-1} \cdots b_0$ of length k.

We use GF(2^k) field in our randomized streaming algorithm for hardness magnification . Let F be a GF(2^k) field (a field of size $q = 2^k$) and has its $R(F) = (2, t_F(u))$. Let $s = a_0 \cdots a_{m-1}$ be a binary string of length m with $m \leq k$, and u be a variable. Define $w(s, u)$ to be the element $\sum_{i=0}^{m-1} a_i u^i$ in GF(2^k). Let x be a string in $\{0, 1\}^*$ and k be an integer at least 1. Let $x = s_{r-1} s_{t-2} \cdots s_1 s_0$ such that each s_i is a substring of x of length k for $i = 1, 2, \cdots, r-1$, and the substring s_0 has its length $|s_0| \leq k$. Each s_i is called a *k-segment* of x for $i = 0, 1, \cdots, r-1$. Define the polynomial $d_x(z) = z^r +$

$\sum_{i=0}^{r-1} w(s_i, u)z^i$, which converts a binary string into a polynomial in GF(2^k).

We develop a streaming algorithm that converts an input string into an element in a finite field. We give the definition to characterize the properties of the streaming algorithm developed in this paper. Our streaming algorithm is to convert an input stream x into an element $d_x(a) \in F = \mathrm{GF}(2^k)$ by selecting a random element a from F.

Definition 1. Let $r_0(n), r_1(n), r_2(n), s(n), u(n)$ be nondecreasing functions from $\mathbb{N}$ to $\mathbb{N}$. Define Streaming($r_0(n), r_1(n), s(n), u(n), r_2(n)$) to be the class of languages L that have one-pass streaming algorithms that has input (n, x) with $n = |x|$ (x is a string and read by streaming), it satisfies

i. It takes $r_0(n)$ time to generate a field $F = \mathrm{GF}(2^k)$, which is represented by $(2, t_F(.))$ with a irreducible polynomial $t_f(.)$ over GF(2) of degree k.

ii. It takes $O(r_1(n))$ random steps before reading the first bit from the input stream x.

iii. It uses $O(s(n))$ space that includes the space to hold the field representation generated by the algorithm. The space for a field representation is $\Omega((\deg(t_F(.)) + 1))$ and $\mathrm{O}((\deg(t_F(.)) + 1))$ for the irreducible polynomial $t_F(.)$ over GF(2).

iv. It takes $O(u(n))$ field conversions to elements in F and $O(u(n))$ field operations in F after reading each bit.

v. It runs $O(r_2(n))$ randomized steps after reading the entire input.

3. Overview of Our Methods

In this section, we give a brief description about our methods used in this paper. Our first result is based on a polynomial method on a finite field whose size affects the hardness of magnification. The second result is a translational method for zero-error probabilistic complexity classes.

3.1. Magnify the Hardness of Sparse Sets

We have a polynomial method over finite fields. Let L be $f(n)$-sparse language in NTIME($t_1(n)$). In order to handle an input string of size n, a finite field $F = \mathrm{GF}(q)$ with $q = 2^k$ for some integer k is selected, and is represented by $R(F) = (2, t_F(z))$, where $t_F(z)$ is a irreducible polynomial over GF(2). An input $y = a_1a_2 \cdots a_n$ is partitioned into k-segments $s_{r-1} \cdots s_1 s_0$ such that each s_i is converted into an element $w(s_i, u)$ in F, and y is transformed into an polynomial $d_y(z) = z^r + \sum_{i=0}^{r-1} w(s_i, u)z^i$. A random element $a \in F$ is chosen in the beginning of streaming algorithm before processing the input stream. The value $d_y(a)$ is evaluated with the procession of input stream. The finite F is large enough such that for different y_1 and y_2 of the same length, $d_{y_1}(.)$ and

$d_{y_2}(.)$ are different polynomials due to their different coefficients derived from y_1 and y_2, respectively. Let $H(y)$ be the set of all $\langle n, a, d_y(a)\rangle$ with $a \in F$ and $n = |y|$. Set $A(n)$ is the union of all $H(y)$ with $y \in L^n$. The set of A is $\cup_{i=1}^{\infty} A(n)$. A small lower bound for the language A is magnified to large lower bound for L.

The size of field F depends on the density of set L and is $\mathrm{O}(f(n)n)$. By the construction of A, if $y \in L$, there are q tuples $\langle n, a, d_y(a)\rangle$ in A that are generated by y via all a in F. For two different y_1 and y_2 of length n, the intersection $H(y_1) \cap H(y_2)$ is bounded by the degree of $d_{y_1}(.)$. If $y \notin L$, the number of items $\langle n, a, d_y(a)\rangle$ generated by y is at most $\frac{q}{4}$ in A. If $y \in L$, the number of items $\langle n, a, d_y(a)\rangle$ generated by y is q in A. This enables us to convert a string x of length n in L into some strings in A of length much smaller than n, make the hardness magnification possible.

3.2. Separation by ZPP-hardness of MCSP

Our another result shows that ZPP-hardness for MCSP implies EXP $\neq$ ZPP. We identify a class of functions that are padding stable, which has the property if $T \in \mathrm{TALLY}(d(n), g(n))$, then $\{1^{n+2^n} : 1^n \in T\} \in \mathrm{TALLY}(d(n), g(n))$. The function pair $d(n) = \log\log n$ and $g(n) = 2^{2^{2n}}$ has this property. We construct a very sparse tally set $L \in \mathrm{EXP} \cap \mathrm{TALLY}(d(n), g(n))$ that separates ZPEXP from ZPP, where ZPEXP is the zero error exponential time probabilistic class. It is based on a diagonal method that is combined with a padding design. A tally language L has a zero-error 2^{2^n}-time probabilistic algorithm implies $L' = \{1^{n+2^n} : 1^n \in L\}$ has a zero-error 2^n-time probabilistic algorithm. Adapting to the method of [11], we prove that if MCSP is $\mathrm{ZPP} \cap \mathrm{TALLY}(d(n), g(n))$-hard under polynomial time truth-table reductions, then EXP $\neq$ ZPP.

4. Hardness Magnification via Streaming

In this section, we show a hardness magnification of sparse sets via a streaming algorithm. A classical algorithm to find irreducible polynomial [15] is used to construct a field that is large enough for our algorithm.

Theorem 2. *[15] There is a deterministic algorithm that constructs a irreducible polynomial of degree n in $\mathrm{O}(p^{\frac{1}{2}}(\log p)^3 n^{3+\epsilon} + (\log p)^2 n^{4+\epsilon})$ operations in F, where F is a finite field $\mathrm{GF}(p)$ with prime number p.*

Definition 3. Let $f(n)$ be a function from N to N. For a language $A \subseteq \{0,1\}^*$, we say A is $f(n)$-sparse if $|A^n| \leq f(n)$ for all large integer n.

4.1. Streaming Algorithm

The algorithm Streaming(.) is based on a language L that is $f(n)$-sparse. It generates a field $F = \mathrm{GF}(2^k)$ and evaluates $d_x(a)$ with a random element a in

F. A polynomial $z^r + \sum_{i=0}^{r-1} b_i z^i = z^r + b_{r-1}z^{r-1} + b_{r-2}z^{r-2} + \cdots + b_0$ can be evaluated by $(\cdots((z + b_{r-1})z + b_{r-2})z + ...)z + b_0$ according to the classical Horner's algorithm. For example, $z^2 + z + 1 = (z + 1)z + 1$.

Algorithm

Streaming(n, x)

Input: an integer n, and string $x = a_1 \cdots a_n$ of the length n;

Steps:

```
Select a field size $q = 2^k$ such that $8f(n)n < q \leq 16f(n)n$.
Generate an irreducible polynomial $t_F(u)$ of degree $k$ over GF(2) such that $(2, t_F(u))$ represents finite $F =$ GF($q$) (by Theorem 2 with $p = 2$);
Let $a$ be a random element in $F$;
Let $r = \lceil \frac{n}{k} \rceil$; (Note that $r$ is the number of $k$-segments of $x$. See Section 2)
Let $j = r - 1$;
Let $v = 1$;
Repeat
{
    Receive the next $k$-segment $s_j$ from the input stream $x$;
    Convert $s_j$ into an element $b_j = w(s_j, u)$ in GF($q$);
    Let $v = v \cdot a + b_j$;
    Let $j = j - 1$;
}
Until $j < 0$ (the end of the stream);
Output $\langle n, a, v \rangle$;
```

End of Algorithm

Now we have our magnification algorithm. Let $M(.)$ be a randomized Turing machine to accept a language A that contains all $\langle |x|, a, d_x(a) \rangle$ with $a \in F$ and $x \in L$. We have the following randomized streaming algorithm to accept L via the randomized algorithm $M(.)$ for A.

Algorithm

Magnification(n, x)

Input integer n and $x = a_1 \cdots a_n$ as a stream;

Steps: Let $y =$Streaming(n, x); Accept if $M(y)$ accepts;

End of Algorithm

4.2. Hardness Magnification

In this section, we derive some results about hardness magnification via sparse set. Our results show a trade off between the hardness magnification and sparseness via the streaming model.

Definition 4. For a nondecreasing function $t(.) : \mathbb{N} \rightarrow \mathbb{N}$, define $\mathrm{BTIME}(t(n))$ the class of languages L that have two-side bounded error probabilistic algorithms with time complexity $O(t(n))$. Define $\mathrm{BPP} = \cup_{c=1}^{\infty}\mathrm{BTIME}(n^c)$.

Theorem 5. *Assume that $u_1(m)$ be nondecreasing function for the time to generate an irreducible polynomial of degree m in $\mathrm{GF}(2)$, and $u_2(m)$ be the nondecreasing function of a time upper bound for the operations $(+, .)$ in $\mathrm{GF}(2^m)$. Let $f(.), t_1(.), t_2(.), t_3(n)$ be nondecreasing functions $\mathbb{N} \rightarrow \mathbb{N}$ with $f(n) \leq 2^{\frac{n}{2}}$, $v(n) = (\log n + \log f(n))$, and $10v(n) + t_1(n) + u_1(10v(n)) + n \cdot u_2(10v(n)) \leq t_2(v(n))$ for all large n. If there is a $f(n)$-sparse set L with $L \in \mathrm{NTIME}(t_1(n))$ and $L \notin \mathrm{Streaming}(u_1(10v(n))), v(n), v(n), 1, t_3(10v(n)))$, then there is a language A such that $A \in \mathrm{NTIME}(t_2(n))$ and $A \notin \mathrm{BTIME}(t_3(n))$.*

Proof: Select a finite field $\mathrm{GF}(q)$ with $q = 2^k$ for an integer k by line 1 of the algorithm streaming(.). For each $x \in L^n$, let x be partitioned into k-segments: $s_{r-1}s_{r-2}\cdots s_0$. Let $w(s_i, u)$ convert s_i into an element of $\mathrm{GF}(q)$ (See Section 2). Define polynomial $d_x(z) = z^r + \sum_{i=0}^{r-1} w(q, s_i)z^i$. For each x, let $H(x)$ be the set $\{\langle n, a, d_x(a)\rangle | a \in \mathrm{GF}(q)\}$, where $n = |x|$. Define set $A(n) = \cup_{y\in L^n} H(y)$ for $n = 1, 2\cdots$, and language $A = \cup_{n=1}^{+\infty} A(n)$.

Claim 1. For any $x \notin L^n$ with $n = |x|$, we have $|H(x) \cap A(n)| < \frac{q}{4}$.
Proof: Assume that for some $x \notin L^n$ with $n = |x|$, $|H(x) \cap A(n)| \geq \frac{q}{4}$. It is easy to see that $r \leq n$ and $k \leq n$ for all large n by the algorithm Streaming(.) and the condition of $f(.)$ in the theorem. Assume that $|H(x) \cap H(y)| < r + 1$ for every $y \in L^n$. Since $A(n)$ is the union $H(y)$ with $y \in L^n$ and $|L^n| \leq f(n)$, there are at most $rf(n) \leq nf(n) < \frac{q}{8}$ elements in $H(x) \cap A(n)$ by line 1 of the algorithm Streaming(.). Thus, $|H(x) \cap A(n)| < \frac{q}{8}$. This brings a contradiction. Therefore, there is a $y \in L^n$ to have $|H(x)\cap H(y)| \geq r+1$. Since the polynomials $d_x(.)$ and $d_y(.)$ are of degrees at most r, we have $d_x(z) = d_y(z)$ (two polynomials are equal). Thus, $x = y$. This brings a contradiction because $x \notin L^n$ and $y \in L^n$.

▮

Claim 2. If $x \in L$, then $\mathrm{Streaming}(|x|, x) \in A$. Otherwise, with probability at most $\frac{1}{4}$, $\mathrm{Streaming}(|x|, x) \in A$.
Proof: For each x, it generates $\langle n, a, d_x(a)\rangle$ for a random $a \in \mathrm{GF}(q)$. Each $a \in \mathrm{GF}(q)$ determines a random path. We have that if $x \in L$, then $\langle n, a, d_x(a)\rangle \in A$, and if $x \notin L$, then $\langle n, a, d_x(a)\rangle \in A$ with probability at most $\frac{1}{4}$ by Claim 1.

▮

Claim 3. $A \in \mathrm{NTIME}(t_2(m))$.
Proof: Let $z = 10v(n) = 10(\log n + \log f(n))$. Each element in field $F = \mathrm{GF}(2^k)$ is of length k. For each $u = \langle n, a, b\rangle$ $(a, b \in F)$, we need to guess a string

$x \in L^n$ such that $b = d_x(a)$. It is easy to see that $v(n) \le |\langle n, a, b\rangle| \le 10v(n)$ for all large n if $\langle n, a, b\rangle \in A$ (See Section 2 about coding). Let $m = |\langle n, a, b\rangle|$. It takes at most $u_1(z)$ steps to generate a irreducible polynomial $t_F(.)$ for the field F by our assumption.

Since $L \in \text{NTIME}(t_1(n))$, checking if $u \in A$ takes nondeterministic $t_1(n)$ steps to guess a string $x \in L^n$, $u_1(z)$ deterministic steps to generate $t_F(u)$ for the field F, $O(z)$ nondeterministic steps to generate a random element $a \in F$, and additional $\text{O}(n \cdot u_2(z))$ steps to evaluate $d_x(a)$ in by following algorithm Streaming(.) and check $b = d_x(a)$. The polynomial $t_F(u)$ in the GF(2) has degree at most z. Each polynomial operation (+ or .) in F takes at most $u_2(z)$ steps. Since $z + t_1(n) + u_1(z) + n \cdot u_2(z) \le t_2(m)$ time by the condition of this theorem, we have $A \in \text{NTIME}(t_2(m))$.

∎

Claim 4. If $A \in \text{BTIME}(t_3(m))$, then $L \in \text{Streaming}(u_1(10v(n)), v(n), v(n), 1, t_3(10v(n)))$.

Proof: The field generated at line 2 in algorithm Streaming(.) takes $u_1(10(\log n + \log f(n)))$ time. Let $x = a_1 \cdots a_n$ be the input string. The string x partitioned into k-segments $s_{r-1} \cdots s_0$. Transform each s_i into an element $b_i = w(s_i, u)$ in GF(q) in the streaming algorithm. We generate a polynomial $d_x(z) = z^r + \sum_{i=0}^{r-1} b_i z^i = z^r + b_{r-1}z^{r-1} + b_{r-2}z^{r-2} + \cdots + b_0$. Given a random element $a \in \text{GF}(q)$, we evaluate $d_x(a) = (\cdots((a+b_{r-1})a+b_{r-2})a+...)a+b_0$ according to the classical algorithm. Therefore, $d_x(a)$ is evaluated in Streaming(.) with input $(|x|, x)$.

If $A \in \text{BTIME}(t_3(m))$, then L has a randomized streaming algorithm that has at most $t_3(10v(n))$ random steps after reading the input, and at most $\text{O}(v(n))$ space. After reading one substring s_i from x, it takes one conversion from a substring of the input to an element of field F by line 10, and at most two field operations by line 11 in the algorithm Streaming(.).

∎

Claim 4 brings a contradiction to our assumption about the complexity of L in the theorem. This proves the theorem. ∎

Proposition 6. *Let $f(n) : \mathbb{N} \to \mathbb{N}$ be a nondecreasing function. If for each fixed $\epsilon \in (0, 1)$, $f(n) \le n^{\epsilon}$ for all large n, then there is a nondecreasing unbounded function $g(n) : \mathbb{N} \to \mathbb{N}$ with $f(n) \le n^{\frac{1}{g(n)}}$.*

Proof: Let $n_0 = 1$. For each $k \ge 1$, let n_k be the least integer such that $n_k \ge n_{k-1}$ and $f(n) \le n^{\frac{1}{k}}$ for all $n \ge n_k$. Clearly, we have the infinite list $n_1 \le n_2 \cdots \le n_k \le \cdots$ such that $\lim_{k \to +\infty} n_k = +\infty$. Define function $g(k) : \mathbb{N} \to \mathbb{N}$ such that $g(n) = k$ for all $n \in [n_{k-1}, n_k)$. For each $n \ge n_k$, we have $f(n) \le n^{\frac{1}{k}}$.

∎

Our Definition 7 is based Proposition 6. It can simplify the proof when we handle a function that is $n^{o(1)}$.

Definition 7. A function $f(n) : \mathbb{N} \to \mathbb{N}$ is $n^{o(1)}$ if there is a nondecreasing function $g(n) : \mathbb{N} \to \mathbb{N}$ such that $\lim_{n\to+\infty} g(n) = +\infty$ and $f(n) \le n^{\frac{1}{g(n)}}$ for all large n. A function $f(n) : \mathbb{N} \to \mathbb{N}$ is $2^{n^{o(1)}}$ if there is a nondecreasing function $g(n) : \mathbb{N} \to \mathbb{N}$ such that $\lim_{n\to+\infty} g(n) = +\infty$ and $f(n) \le 2^{n^{\frac{1}{g(n)}}}$ for all large n.

Corollary 8. *If there exists a $2^{n^{o(1)}}$-sparse language L in* $\mathrm{NTIME}(2^{n^{o(1)}})$ *such that L does not have any randomized streaming algorithm with $n^{o(1)}$ updating time, and $n^{o(1)}$ space, then* $\mathrm{NEXP} \neq \mathrm{BPP}$.

Proof: Let $g(n) : \mathbb{N} \to \mathbb{N}$ be an arbitrary unbounded nondecreasing function that satisfies $\lim_{n\to+\infty} g(n) = +\infty$ and $g(n) \le \log\log n$. Let $t_1(n) = f(n) = 2^{n^{\frac{1}{g(n)}}}$ and Let $t_2(n) = 2^{2n}$, $t_3(n) = n^{\sqrt{g(n)}}$, and $v(n) = (\log n + \log f(n))$.

It is easy to see that $v(n) = n^{\mathrm{o}(1)}$, and both $u_1(n)$ and $u_2(n)$ are $n^{\mathrm{O}(1)}$ (see Theorem 2). For any fixed $c_0 > 0$, we have $t_2(v(n)) > t_2(\log f(n)) \ge t_2(n^{\frac{1}{g(n)}}) > t_1(n) + n^{c_0}$ for all large n. For all large n, we have

$$\begin{aligned} t_3(10v(n)) &\le t_3(20\log f(n)) = t_3(20n^{\frac{1}{g(n)}}) && (1)\\ &\le (20n^{\frac{1}{g(n)}})^{\sqrt{g(20n^{\frac{1}{g(n)}})}} \le (n^{\frac{2}{g(n)}})^{\sqrt{g(n)}} = n^{o(1)}. && (2) \end{aligned}$$

Clearly, these functions satisfy the inequality of the precondition in Theorem 5. Assume $L \in \mathrm{Streaming}(\mathrm{poly}(v(n)), v(n), v(n), 1, t_3(10v(n)))$. With $O(v(n)) = n^{o(1)}$ space, we have a field representation $(2, t_F(.))$ with $\deg(t_F(.)) = n^{o(1)}$. Thus, each field operation takes $n^{o(1)}$ time by the brute force method for polynomial addition and multiplication. We have $t_3(10v(n)) = n^{o(1)}$ by inequality (2). Thus, the streaming algorithm updating time is $n^{o(1)}$. Therefore, we have that L has a randomized streaming algorithm with $n^{o(1)}$ updating time, and $n^{o(1)}$ space. This gives a contradiction. So,
$L \notin \mathrm{Streaming}(\mathrm{poly}(v(n)), v(n), v(n), 1, t_3(10v(n)))$. By Theorem 5, there is $A \in \mathrm{NTIME}(t_2(n))$ such that $A \notin \mathrm{BTIME}(t_3(n))$. Therefore, $A \notin \mathrm{BPP}$. Thus, $\mathrm{NEXP} \neq \mathrm{BPP}$. ▮

5. Implication of ZPP-Hardness of MCSP

In this section, we show that if MCSP is ZPP∩TALLY-hard, then EXP ≠ ZPP. The conclusion still holds if TALLY is replaced by a very sparse subclass of TALLY languages.

Definition 9. For a nondecreasing function $t(.) : \mathbb{N} \to \mathbb{N}$, define $\mathrm{ZTIME}(t(n))$ the class of languages L that have zero-error probabilistic algorithms with time complexity $O(t(n))$. Define $\mathrm{ZPP} = \cup_{c=1}^{\infty}\mathrm{ZTIME}(n^c)$, and
$\mathrm{ZPEXP} = \cup_{c=1}^{\infty}\mathrm{ZTIME}(2^{n^c})$.

Definition 10. For an nondecreasing function $f(n) : \mathbb{N} \to \mathbb{N}$, define $\text{TALLY}[f(k)]$ to be the class of tally set $A \subseteq \{1\}^*$ such that for each $1^m \in A$, there is an integer $i \in \mathbb{N}$ with $m = f(i)$. For a tally language $T \subseteq \{1\}^*$, define $\text{Pad}(T) = \{1^{2^n+n} | 1^n \in T\}$.

Definition 11. For two languages A and B, a polynomial time *truth-table reduction* from A to B is a polynomial time computable function $f(.)$ such that for each instance x for A, $f(x) = (y_1, \cdots, y_m, C(.))$ to satisfy $x \in A$ if and only if $C(B(y_1), \cdots, B(y_m)) = 1$, where $C(.)$ is circuit of m input bits and $B(.)$ is the characteristic function of B.

Let $\leq_r^P$ be a type of polynomial time reductions ($\leq_{tt}^P$ represents polynomial time truth-table reductions), and C be a class of languages. A language A is C-hard under $\leq_r^P$ reductions if for each $B \in C$, $B \leq_r^P A$.

Definition 12. Let k be an integer. Define two classes of functions with recursions: (1) $\log^{(1)}(n) = \log_2 n$, and $\log^{(k+1)}(n) = \log_2(\log^{(k)}(n))$. (2) $\exp^{(1)}(n) = 2^n$, and $\exp^{(k+1)}(n) = 2^{\exp^{(k)}(n)}$.

Definition 13. For two nondecreasing functions $d(n), g(n) : \mathbb{N} \to \mathbb{N}$, the pair $(d(n), g(n))$ is *time constructible* if $(d(n), g(n))$ can be computed in time $d(n) + g(n)$ steps.

Definition 14. Define $\text{TALLY}(d(n), g(n))$ to be the class of tally sets T such that $|T^{\leq n}| \leq d(n)$ and for any two strings $1^n, 1^m \in T$ with $n < m$, they satisfy $g(n) < m$. We call $d(n)$ to be the *density function* and $g(n)$ to be the *gap function*. A gap function $g(n)$ is *padding stable* if $g(2^n + n) < 2^{g(n)} + g(n)$ for all $n > 1$.

Lemma 15.

i. Assume the gap function $g(n)$ is padding stable. If $T \in \text{TALLY}(d(n), g(n))$, then $\text{Pad}(T) \in \text{TALLY}(d(n), g(n))$.

ii. For each integer $k > 0$, $g(n) = \exp^{(k)}(2n)$ is padding stable.

Proof: Part i. Let 1^n be a string in T. The next shortest string $1^m \in T$ with $n < m$ satisfies $g(n) < m$. We have 1^{2^n+n} and 1^{2^m+m} are two consecutive neighbor strings in $\text{Pad}(T)$ such that there is no other string $1^k \in \text{Pad}(T)$ with $2^n + n < k < 2^m + m$. We have $g(2^n + n) < 2^{g(n)} + g(n) < 2^m + m$. Since the strings in $\text{Pad}(T)^{\leq n}$ are one-one mapped from the strings in T with length less than n, $|\text{Pad}(T)^{\leq n}| \leq |T^{\leq n}| \leq d(n)$, we have $\text{Pad}(T) \in \text{TALLY}(d(n), g(n))$. This proves Part (i).

Part ii. We have inequality $g(2^n+n) = \exp^{(k)}(2(2^n+n)) < \exp^{(k)}(4 \cdot 2^n) = \exp^{(k)}(2^{n+2}) \leq \exp^{(k)}(2^{2n}) = 2^{g(n)} < 2^{g(n)} + g(n)$. Therefore, gap function $g(n)$ is padding stable. This proves Part ii. ∎

Lemma 16. *Let $d(n)$ and $g(n)$ be nondecreasing unbounded functions from N to N, and $(d(n), g(n))$ is time constructible. Then there exists a time constructible increasing unbounded function $f(n) : \mathbb{N} \to \mathbb{N}$ such that* $\mathrm{TALLY}[f(n)] \subseteq \mathrm{TALLY}(d(n), g(n))$.

Proof: Compute the least integer n_1 with $d(n_1) > 0$. Let s_1 be the number of steps for the computation. Define $f(1) = \max(s_1, n_1)$. Assume that $f(k-1)$ has been defined. We determine the function value $f(k)$ below.

For an integer $k > 0$, compute $g(f(k-1))$ and the least k numbers $n_1 < n_2 < \cdots < n_k$ such that $0 < d(n_1) < d(n_2) < \cdots < d(n_k)$. Assume the computation above takes s steps. Define $f(k)$ to be the $\max(2s, n_k, g(f(k-1)) + 1)$. For each language $T \in \mathrm{TALLY}[f(n)]$, there are at most k strings in T with length at most $f(k)$. On the other hand, $d(n_k) \geq k$ by the increasing list $0 < d(n_1) < d(n_2) < \cdots < d(n_k)$. Therefore, we have $|T^{\leq n_k}| \leq k \leq d(n_k)$. Furthermore, we also have $g(f(k-1)) < f(k)$. Since s is the number of steps to determine the values s, n_k, and $g(f(k-1)) + 1$. We have $2s \leq f(k)$. Thus, $f(k)$ can be computed in $f(k)$ steps by spending some idle steps. Therefore, the function $f(.)$ is time constructible. ▮

We will use the notion $\mathrm{TALLY}[f(k)]$ to characterize extremely sparse tally sets with fast growing function such as $f(k) = 2^{2^{2^k}}$. It is easy to see that $\mathrm{TALLY} = \mathrm{TALLY}[I(.)]$, where $I(.)$ is the identity function $I(k) = k$.

Lemma 17. *Let $d(n)$ and $g(n)$ be nondecreasing unbounded functions. If function $g(n))$ is padding stable, then there is a language A such that $A \in \mathrm{ZTIME}(2^{O(n)}) \cap \mathrm{TALLY}(d(n), g(n))$ and $A \notin \mathrm{ZPP}$.*

Proof: It is based on the classical translational method. Assume $\mathrm{ZTIME}(2^{O(n)}) \cap \mathrm{TALLY}(d(n), g(n)) \subseteq \mathrm{ZPP}$. Let $f(.)$ be a time constructible increasing unbounded function via Lemma 16 such that $\mathrm{TALLY}[f(n)] \subseteq \mathrm{TALLY}(d(n), g(n))$. Let $t_1(n) = 2^{2^n}$ and $t_2(n) = 2^{2^{n-1}}$. Let L be a tally language in $\mathrm{DTIME}(t_1(n)) \cap \mathrm{TALLY}[f(n)]$, but it is not in $\mathrm{DTIME}(t_2(n))$. Such a language L can be constructed via a standard diagonal method. Let $M_1, \cdots, M_2$ be the list of Turing machines such that each M_i has time upper bound by function $t_2(n)$. Define language $L \in \mathrm{TALLY}[f(n)]$ such that for each k, $1^{f(k)} \in L$ if and only if $M_k(1^{f(k)})$ rejects in $t_2(f(k))$ steps. We have $L \in \mathrm{TALLY}(d(n), g(n))$ by Lemma 16.

Let $L_1 = \mathrm{Pad}(L)$. We have $L_1 \in \mathrm{TALLY}(d(n), g(n))$ by Lemma 15. We have $L_1 \in \mathrm{DTIME}(2^{O(n)}) \subseteq \mathrm{ZTIME}(2^{O(n)})$. Thus, $L_1 \in \mathrm{ZPP}$. So, $L \in \mathrm{ZTIME}(2^{O(n)})$. Therefore, $L \in \mathrm{ZTIME}(2^{O(n)}) \cap \mathrm{TALLY}(d(n), g(n))$. We have $L \in \mathrm{ZPP}$. Thus, $L \in \mathrm{DTIME}(2^{n^{O(1)}}) \subseteq \mathrm{DTIME}(2^{2^{n-1}})$. This brings a contradiction. ▮

Theorem 18. *Let $d(n)$ and $g(n)$ be nondecreasing unbounded functions from $\mathbb{N}$ to $\mathbb{N}$. Assume that $g(n)$ is padding stable. If* MCSP *is* $\mathrm{ZPP} \cap \mathrm{TALLY}(d(n), g(n))$*-hard under polynomial time truth-table reductions, then* EXP$\neq$ ZPP.

Proof: Assume that MCSP is $(\mathrm{ZPP}\cap \mathrm{TALLY}(d(n), g(n))$-hard under polynomial time truth-table reductions, and EXP = ZPP.

Let L be a language in $\mathrm{ZTIME}(2^{O(n)})\cap \mathrm{TALLY}[d(.), g(.)]$, but $L\notin \mathrm{ZPP}$ by Lemma 17. Let $L'=\mathrm{Pad}(L)$. Clearly, every string 1^y in L' has the property that $y=2^n+n$ for some integer n. This property is easy to check and we reject all strings without this property in linear time. We have $L'\in \mathrm{ZPP}$. Therefore, there is a polynomial time truth-table reduction from L' to MCSP via a polynomial time truth-table reduction $M(.)$. Let polynomial $p(n)=n^c$ be the running time for $M(.)$ for a fixed c and $n\geq 2$.

Define the language $R=\{(1^n, i, j)$, the i-th bit of j-th query of $M(1^{n+2^n})$ is equal to 1, and $i, j\leq p(n+2^n)\}$. We can easily prove that R is in EXP. Therefore, $R\in \mathrm{ZPP}\subseteq \mathrm{P/poly}$ (See [1]).

Therefore, there is a class of polynomial size circuits $\{C_n\}_{n=1}^{\infty}$ to recognize R such that $C_n(.)$ recognize all $(1^n, i, j)$ with $i, j\leq p(n+2^n)$ in R. Assume that the size of C_n is of size at most $q(n)=n^{t_0}+t_0$ for a fixed t_0. For an instance $x=1^n$ for L, consider the instance $y=1^{n+2^n}$ for L'. We can compute all non-adaptive queries $\langle T, s(n)\rangle$ to MCSP in $2^{n^{O(1)}}$ time via $M(y)$. If $s(n)\geq q(n)$, the answer from MCSP for the query $\langle T, s(n)\rangle$ is yes since $\langle T, s(n)\rangle$ can be generated as one of the instances via the circuit $C_n(.)$. If $s(n)<q(n)$, we can use a brute force method to check if there exists a circuit of size at most $q(n)$ to generate T. It takes $2^{n^{O(1)}}$ time. Therefore, $L\in \mathrm{EXP}$. Thus, $L\in \mathrm{ZPP}$. This bring a contradiction as we already assume $L\notin \mathrm{ZPP}$. ▌

Corollary 19. *For any integer k, if* MCSP *is* $\mathrm{ZPP}\cap\mathrm{TALLY}(\log^{(k)}(n), \exp^{(k)}(2n))$-*hard under polynomial time truth-table reductions, then* EXP≠ ZPP.

Proof: It follows from Theorem 18 and Lemma 15. ▌

Corollary 20. *For any integer k, if* MCSP *is* $\mathrm{ZPP}\cap\mathrm{TALLY}$-*hard under polynomial time truth-table reductions, then* EXP≠ ZPP*.*

Corollary 21. *If* MCSP *is* ZPP-*hard under polynomial time truth-table reductions, then* EXP≠ ZPP*.*

6. Conclusions

In this paper, we develop an algebraic method to magnify the hardness of sparse sets in nondeterministic classes via a randomized streaming model. It has a flexible trade off between the sparseness and time complexity. This shows connection to the major problems to prove NEXP ≠ BPP. We also prove that if MCSP is ZPP-hard, then EXP ≠ ZPP.

Acknowledgements: This research was supported in part by National Science Foundation Early Career Award 0845376, and Bensten Fellowship of the University of Texas Rio Grande Valley. Part of this research was conducted while the author was visiting the School of Computer Science and Technology of Hengyang Normal University in the summer of 2019 and was supported by National Natural Science Foundation of China 61772179.

References

[1] L. Adleman. Two theorems on random polynomial time. In *Proceedings of the 19th Annual IEEE Symposium on Foundations of Computer Science*, pages 75–83, 1978.

[2] E. Allender and S. Hirahara. New insights on the (non-)hardness of circuit minimization and related problems. In *42nd International Symposium on Mathematical Foundations of Computer Science, MFCS 2017, August 21-25, 2017 - Aalborg, Denmark*, pages 54:1–54:14, 2017.

[3] E. Allender, D. Holden, and V. Kabanets. The minimum oracle circuit size problem. *Computational Complexity*, 26(2):469–496, 2017.

[4] L. Chen, C. Jin, and R. Williams. Hardness magnification for all sparse NP languages. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:118, 2019.

[5] S. Hirahara, I. C. Oliveira, and R. Santhanam. Np-hardness of minimum circuit size problem for OR-AND-MOD circuits. In *33rd Computational Complexity Conference, CCC 2018, June 22-24, 2018, San Diego, CA, USA*, pages 5:1–5:31, 2018.

[6] S. Hirahara and R. Santhanam. On the average-case complexity of MCSP and its variants. In *32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia*, pages 7:1–7:20, 2017.

[7] S. Hirahara and O. Watanabe. Limits of minimum circuit size problem as oracle. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 18:1–18:20, 2016.

[8] J. M. Hitchcock and A. Pavan. On the np-completeness of the minimum circuit size problem. In *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, pages 236–245, 2015.

[9] T. Hungerford. *Algebra.* Springer-Verlag, 1974.

[10] D. M. McKay, C. D. Murray, and R. R. Williams. Weak lower bounds on resource-bounded compression imply strong separations of complexity classes. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 1215–1225, 2019.

[11] C. D. Murray and R. R. Williams. On the (non) np-hardness of computing circuit complexity. *Theory of Computing*, 13(1):1–22, 2017.

[12] I. C. Oliveira, J. Pich, and R. Santhanam. Hardness magnification near state-of-the-art lower bounds. In *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA.*, pages 27:1–27:29, 2019.

[13] I. C. Oliveira and R. Santhanam. Hardness magnification for natural problems. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 65–76, 2018.

[14] U. Schöning. A low and a high hierarchy within NP. *JCSS*, 27:14–28, 1983.

[15] V. Shoup. New algorithms for finding irreducible polynomials over finite fields. In *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, pages 283–290, 1988.